Package 'constellation'

October 12, 2022

Type Package

Title Identify Event Sequences Using Time Series Joins

Version 0.2.0

Date 2018-03-23

Depends R (>= 3.0.0)

Imports data.table (>= 1.9.5)

Suggests knitr, rmarkdown, ggplot2, devtools, testthat

Description Examine any number of time series data frames to identify instances in which various criteria are met within specified time frames. In clinical medicine, these types of events are often called ``constellations of signs and symptoms", because a single condition depends on a series of events occurring within a certain amount of time of each other. This package was written to work with any number of time series data frames and is optimized for speed to work well with data frames with millions of rows.

License GPL (>= 2)

LazyData true

URL https://github.com/marksendak/constellation

RoxygenNote 6.0.1

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation no

Author Mark Sendak [aut, cre]

Maintainer Mark Sendak <mark.sendak@gmail.com>

Repository CRAN

Date/Publication 2018-03-27 03:15:25 UTC

R topics documented:

`bundle` *Identify bundle items that occur around a given event*

Description

A function that reads in a data frame of incident events along with multiple time series data frames of bundle items and calculates whether or not each bundle item occurs within a defined time window around the incident event. The user must provide names for each bundle item, define the time window around the incident events to consider, a name for the incident event, and variables to use to join the different tables. Lastly, the user can specify whether to return all instances that each bundle item occurs around the incident event, or whether to pull only the first or last instant for each bundle item. All time series data frames must contain columns for joining the tables (join_key) and time stamps (time_var). The time_var column must be class POSIXct in all the data frames. This function can ingest an arbitrary number of data frames for different bundle items around an incident event.

Usage

```
bundle(events, ..., bundle_names, window_hours_pre, window_hours_post, join_key,
  time_var, event_name, mult = c("all", "first", "last"))
```

Arguments

`events`	A time series data frame of incident events. The bundle events are searched for around a given time window of these events. The events data frame must include the columns 'join_key' and 'time_var'
`...`	An arbitrary number of time series data frames that each include the columns 'join_key' and 'time_var'. Each data frame consists of a bundle item that is important to find around the specified events.
`bundle_names`	A vector of strings specifying the name of each event. The order of strings in the vector should align with the order of data frames passed in '...'.
`window_hours_pre`	A single numeric or vector of numerics speficying . the number of hours before the events in the events data frame that each bundle item is considered relevant.

If a single numeric is passed, that time window before the events is applied to all bundle items.

`window_hours_post`
: A single numeric or vector of numerics speficying . the number of hours after the events in the events data frame that each bundle item is considered relevant. If a single numeric is passed, that time window after the events is applied to all bundle items.

`join_key`
: A string name of the column to join all time series data frames

`time_var`
: A string name of the time stamp column in all time series data frames. The class of time_var must be POSIXct in all data frames.

`event_name`
: A string name of the events in the events data frame

`mult`
: A string specifying whether to return the first, last, or all instance(s) of every bundle item occurring within the specified time window of events. The default value is all.

Value

A data.frame, data.table with a time stamp for every event of interest, columns for the start and end of the time window of interest, and columns for every bundle item. The value in bundle item columns is the timestamp (time_var) that the bundle item is observed within the given window.

Imported functions

foverlaps() from data.table and general data.table syntax

Errors

This function returns errors for:

- missing arguments (only the mult argument has a default value)
- passing arguments with invalid classes (events and bundle items must be data frames, bundle_names must be a string, window_hours_pre and window_hours_post must be numerics, and event_name must be a string)
- passing an invalid mult value
- passing join_key or time_var values that are not column names in all time series data frames
- passing an invalid number of window_hours_pre or window_hours_post values (1 or the number of bundle data frames).

Examples

```
library(data.table)
temp <- as.data.table(vitals[VARIABLE == "TEMPERATURE"])
pulse <- as.data.table(vitals[VARIABLE == "PULSE"])
resp <- as.data.table(vitals[VARIABLE == "RESPIRATORY_RATE"])

temp[, RECORDED_TIME := as.POSIXct(RECORDED_TIME,
  format = "%Y-%m-%dT%H:%M:%SZ", tz = "UTC")]
pulse[, RECORDED_TIME := as.POSIXct(RECORDED_TIME,
```

```
  format = "%Y-%m-%dT%H:%M:%SZ", tz = "UTC")]
resp[, RECORDED_TIME := as.POSIXct(RECORDED_TIME,
  format = "%Y-%m-%dT%H:%M:%SZ", tz = "UTC")]

# Pass single window_hours_pre
# All instances of bundle items within time window of event
bundle(temp, pulse, resp,
    bundle_names = c("PLATELETS", "INR"), window_hours_pre = 24,
    window_hours_post = c(6, 6), join_key = "PAT_ID",
    time_var = "RECORDED_TIME", event_name = "CREATININE", mult = "all")
# Pass different window_hours_pre for each bundle time series data frame
# All instances of bundle items within time window of event
bundle(temp, pulse, resp,
    bundle_names = c("PLATELETS", "INR"), window_hours_pre = c(24, 12),
    window_hours_post = c(6, 6), join_key = "PAT_ID",
    time_var = "RECORDED_TIME", event_name = "CREATININE", mult = "all")
# Pass different window_hours_pre for each bundle time series data frame
# First instance of each bundle item within time window of event
bundle(temp, pulse, resp,
    bundle_names = c("PLATELETS", "INR"), window_hours_pre = c(24, 12),
    window_hours_post = c(6, 6), join_key = "PAT_ID",
    time_var = "RECORDED_TIME", event_name = "CREATININE", mult = "first")
# Pass different window_hours_pre for each bundle time series data frame
# Last instance of each bundle item within time window of event
bundle(temp, pulse, resp,
    bundle_names = c("PLATELETS", "INR"), window_hours_pre = c(24, 12),
    window_hours_post = c(6, 6), join_key = "PAT_ID",
    time_var = "RECORDED_TIME", event_name = "CREATININE", mult = "last")
```

`constellate` *Identify when a constellation of events occur*

Description

A function that reads in multiple time series data frames and calculates instances when a constellation of events occur. The user must specify the number of hours over which each event must take place, a variable to use to join the tables, and the time stamp variable. The timestamps variable in every data frame must be POSIXct class. In addition, the user must specify the event name and whether to keep all instances that events occur, or only the first or last instance. This function can ingest an arbitrary number of data frames with longitudinal time series data.

Usage

```
constellate(..., window_hours, join_key, time_var, event_name, mult = c("all",
  "first", "last"))
```

Arguments

...	An arbitrary number of time series data frames that each include the columns 'join_key' and 'time_var'
window_hours	A single numeric or vector of numerics specifying the number of hours to search for each event. The order of numerics in the vector should align with the order of data frames passed in '...'.
join_key	A string name of the column to join all time series data frames
time_var	A string name of the time stamp column in all time series data frames. The class of time_var must be POSIXct in all data frames.
event_name	A string name for events across the time series data frames
mult	A string specifying whether to return the first, last, or all instance(s) with a default value of all

Value

A data.frame, data.table with time stamps of qualifying events.

Imported functions

general data.table syntax

Errors

This function returns errors for:

- missing arguments (no arguments have defaults)
- passing an invalid mult value
- passing arguments with invalid classes (window_hours must be numeric and event_name must be a string)
- passing join_key or time_var values that are not column names in all time series data frames
- passing an invalid number of window_hours values (1 or the number of event data frames).

Examples

```
library(data.table)
temp <- as.data.table(vitals[VARIABLE == "TEMPERATURE"])
pulse <- as.data.table(vitals[VARIABLE == "PULSE"])
resp <- as.data.table(vitals[VARIABLE == "RESPIRATORY_RATE"])
wbc <- as.data.table(labs[VARIABLE == "WBC"])
temp[, RECORDED_TIME := as.POSIXct(RECORDED_TIME,
  format = "%Y-%m-%dT%H:%M:%SZ", tz = "UTC")]
pulse[, RECORDED_TIME := as.POSIXct(RECORDED_TIME,
  format = "%Y-%m-%dT%H:%M:%SZ", tz = "UTC")]
resp[, RECORDED_TIME := as.POSIXct(RECORDED_TIME,
  format = "%Y-%m-%dT%H:%M:%SZ", tz = "UTC")]
wbc[, RECORDED_TIME := as.POSIXct(RECORDED_TIME,
  format = "%Y-%m-%dT%H:%M:%SZ", tz = "UTC")]
```

```
# Pass single time window for all time series data frames
# Subset first event
constellate(temp, pulse, resp, window_hours = 6, join_key = "PAT_ID",
 time_var = "RECORDED_TIME", event_name = "sirs_vitals", mult = "first")
# Pass different time window for each time series data frame
# Subset first event
constellate(temp, pulse, resp, wbc, window_hours = c(6,6,6,24),
 join_key = "PAT_ID", time_var = "RECORDED_TIME",
 event_name = "SEPSIS", mult = "first")
# Pass different time window for each time series data frame
# Identify all events
constellate(temp, pulse, resp, wbc, window_hours = c(6,6,6,24),
 join_key = "PAT_ID", time_var = "RECORDED_TIME",
 event_name = "SEPSIS", mult = "all")
```

`constellate_criteria` *Provide details about individual events within a constellation*

Description

A function that reads in multiple time series data frames for various events and builds indicator variables for each event. Individual events that occur within a specified number from a timestamp are flagged. The variables for each event can be populated with the time the event took place, a boolean variable (0 or 1) indicating whether or not the event took place, or the result of the variable at the time the event took place.

Usage

```
constellate_criteria(..., criteria_names, window_hours, join_key, time_var,
  value = c("boolean", "time", "result"), result_var = NULL)
```

Arguments

`...`	An arbitrary number of time series data frames that each include the columns 'join_key' and 'time_var'
`criteria_names`	A vector of strings specifying the name of each event. The order of strings in the vector should align with the order of data frames passed in '...'.
`window_hours`	A single numeric or vector of numerics specifying the number of hours to search for each event. The order of numerics in the vector should align with the order of data frames passed in '...'.
`join_key`	A string name of the column to join all time series data frames
`time_var`	A string name of the time stamp column in all time series data frames. The class of time_var must be POSIXct in all data frames.

`value` A string specifying the value to be entered within each criteria column. Options include boolean (0 or 1, depdending on whether the criteria event occurred), the time of the criteria event, or the result stored within the criteria event. The default value is boolean.

`result_var` A string name of the value variable in all data frames. This argument should only be supplied if the "result" option is selected in the value argument.

Details

The user passes an arbitrary number of time series data frames and specifies a name and number of hours to search for each event. The user must also specify a variable to use to join the tables, and the time stamp variable. The timestamps variable in every data frame must be POSIXct class. Finally, the user selects how to populate the individual event variables.

This function extends the constellate function to address a different set of questions, including: 1) at a specific timestamp, which events do and do not occur? 2) what is the sequence of events that trigger the constellation of events that I'm interested in? 3) What are the results of each criteria at the times that each criteria are met? This function can be used to calculate risk scores at any measurement timestamp by building a new variable after the function runs and returns the new data frame. The risk score can add up the crieteria from boolean values (e.g. SIRS Criteria) or can be a linear combination of criteria (e.g., NEWS Score).

Value

A data.frame, data.table with indicator variables for each event. The total number of rows is the unique number of time stamps for all combined measurements.

Imported functions

general data.table syntax

Errors

This function returns errors for:

- missing arguments (value has a default argument and result_var is missing by default)
- passing a window_hours value that is not numeric
- passing join_key or time_var values that are not column names in all time series data frames
- passing an invalid number of criteria_names (must be equal to number of criteria event data frames)
- passing an invalid number of window_hours values (1 or the number of criteria event data frames).

Examples

```
library(data.table)
temp <- as.data.table(vitals[VARIABLE == "TEMPERATURE"])
pulse <- as.data.table(vitals[VARIABLE == "PULSE"])
resp <- as.data.table(vitals[VARIABLE == "RESPIRATORY_RATE"])
```

```
temp[, RECORDED_TIME := as.POSIXct(RECORDED_TIME,
  format = "%Y-%m-%dT%H:%M:%SZ", tz = "UTC")]
pulse[, RECORDED_TIME := as.POSIXct(RECORDED_TIME,
  format = "%Y-%m-%dT%H:%M:%SZ", tz = "UTC")]
resp[, RECORDED_TIME := as.POSIXct(RECORDED_TIME,
  format = "%Y-%m-%dT%H:%M:%SZ", tz = "UTC")]

# Pass single window_hours
constellate_criteria(temp, pulse, resp, criteria_names = c("TEMPERATURE",
 "PULSE", "RESPIRATORY_RATE"), window_hours = 6, join_key = "PAT_ID",
 time_var = "RECORDED_TIME", value = "time")
# Pass vector for window_hours
constellate_criteria(temp, pulse, resp, criteria_names = c("TEMPERATURE",
 "PULSE", "RESPIRATORY_RATE"), window_hours = c(6,6,6), join_key = "PAT_ID",
 time_var = "RECORDED_TIME", value = "time")
# Show the value of each criteria at the time the event occurs
constellate_criteria(temp, pulse, resp, criteria_names = c("TEMPERATURE",
 "PULSE", "RESPIRATORY_RATE"), window_hours = c(6,6,6), join_key =
 "PAT_ID", time_var = "RECORDED_TIME", value = "result",
 result_var = "VALUE")
```

`incidents` *Identify incident events separated by a minimum time window*

Description

A function that reads in a time series data frame along with a specified time window and identifies incident events that are separated in time. The user must specify the number of hours over which events are considered to be the same episode, the time stamp variable, and an optional variable to group episodes. This function was motivated by examples where there may be multiple observations of the same illness episode combined with observations of distinct illness episodes and there is a need to distinguish between episodes. This function assumes that the duration of an episode is non-zero. If every non-equal instant is a distinct episode, there is no need to use this function. Two ways to distinguish between episodes with non-zero duration over time are: (1) bucket observations over a pre-fixed time frame that is applied to all observation; (2) bucket observations over a fixed window of time. An example of (1) is considering observations in the same month to be the same episode and observations in different months to be distinct episodes. The incident() function addresses (2) by specifying a time window and identifying the first observation of each episode. The use can also specify a 'join_key' variable (person, encounter, etc.) to group episodes. The 'window_hours' argument serves as the lower bound to separate observations that are considered the same episode versus distinct episodes.

Usage

```
incidents(data, window_hours, time_var, join_key = NULL)
```

Arguments

data	A time series data frame that includes the columns 'join_key' and 'time_var'
window_hours	A numeric value specifying the number of hours to separate contiguous episodes and distinct episodes
time_var	A string name of the time stamp column in the time series data frame
join_key	An optional string name of the column to group observations

Value

A data.frame, data.table with the time stamps of distinct, incident episodes separated by at least 'window_hours'

Imported functions

general data.table syntax

Errors

This function returns errors for:

- missing arguments (join_key is missing by default)
- passing arguments with invalid classes (data must be a data frame and window_hours must be numeric)
- passing join_key or time_var values that are not column names in input data
- passing time_var column in data that is not POSIXct class

Examples

```
library(data.table)
systolic_bp <- as.data.table(vitals[VARIABLE == "SYSTOLIC_BP"])
systolic_bp[, RECORDED_TIME := as.POSIXct(RECORDED_TIME,
  format = "%Y-%m-%dT%H:%M:%SZ", tz = "UTC")]

# Identify systolic blood pressure measurements for each patient that are
 # separated by at least 24 hours
incidents(systolic_bp, window_hours = 24, join_key = "PAT_ID",
 time_var = "RECORDED_TIME")

# Identify systolic blood pressure measurements that are separated by at
 # least 24 hours
incidents(systolic_bp, window_hours = 24, time_var = "RECORDED_TIME")
```

labs *Synthesized lab results for cohort of 100 synthetic patients*

Description

A dataset containing 3,150 lab results for 96 unique synthetic patients. There are 6 unique labs, including platelets, serum creatinine, international normalized ratio (INR), white blood count (WBC), serum lactate, and serum bilirubin. Patient-level sampling rate, mean numeric value, standard deviation, and number of measurements per patient follow patterns observed in real electronic health record data. This dataset is included to demonstrate sepsis identification.

Usage

```
labs
```

Format

A data frame with 3150 rows and 4 variables:

PAT_ID patient identification number, randomly generated integer

RECORDED_TIME recorded datetime of lab measurement, character

VALUE lab result value, numeric

VARIABLE categorical variable specifying the lab name, character

Source

Randomly synthesized patient data

Examples

```
## Not run:
 labs

## End(Not run)
```

orders *Synthesized blood culture orders for cohort of 100 synthetic patients*

Description

A dataset containing 59 blood culture orders for 27 unique synthetic patients. There is only 1 order for blood cultures. Patient-level sampling rate, mean numeric value, standard deviation, and number of measurements per patient follow patterns observed in real electronic health record data. This dataset is included to demonstrate sepsis identification.

Usage

```
orders
```

Format

A data frame with 59 rows and 3 variables:

PAT_ID patient identification number, randomly generated integer

ORDER_TIME order datetime of blood culture, character

VARIABLE categorical variable specifying the blood culture order name, character

Source

Randomly synthesized patient data

Examples

```
## Not run:
 orders

## End(Not run)
```

---

value_change *Identify changes in a value over time*

Description

A function that reads in a time series data frame along with a specified value change and identifies instances where the value change occurs. The user must specify the number of hours over which the value change must take place, the magnitude and direction of the value change, a variable to use to join the table to itself, the time stamp variable, and the value variable. The timestamps variable in every data frame must be POSIXct class. The user must also specify whether to keep all instances that the value change occurs, or only the first or last instance. This function must be used carefully, because certain types of arguments will cause the function to output a data frame with nrow(data)^2, where 'data' is the input data. More specifically, if the user is trying to detect small variations in a value over a large period of time, the size of input 'data' should be limited.

Usage

```
value_change(data, value, direction = c("all", "up", "down"), window_hours,
  join_key, time_var, value_var, mult = c("all", "first", "last"))
```

Arguments

data	A time series data frame that includes the columns 'join_key', 'time_var', and 'value_var'
value	A numeric value specifying the magnitude of change to identify
direction	A string value specifying whether to identify changes in the value up (an increase), down (a decrease), or all (both). The default value is all.
window_hours	A numeric value specifying the number of hours to search for the value change
join_key	A string name of the column to join the time series data frame to itself. In other words, the primary key to the 'data' argument.
time_var	A string name of the time stamp column in all time series data frames. The class of time_var must be POSIXct in all data frames.
value_var	A string name of the value variable column in the time series data frame
mult	A string specifying whether to return the first, last, or all instance(s) of the value change with a default value of all

Value

A data.frame, data.table with time stamps of value changes over time along with values and time stamps for prior measurements

Imported functions

foverlaps() from data.table and general data.table syntax

Errors

This function returns errors for:

- missing arguments (no arguments have defaults)
- passing an invalid direction or mult value
- passing arguments with invalid classes (data must be a data frame, value must be numeric, and window_hours must be numeric)
- passing join_key, time_var, or value_var values that are not column names in input data frames

Examples

```
library(data.table)
systolic_bp <- as.data.table(vitals[VARIABLE == "SYSTOLIC_BP"])

systolic_bp[, RECORDED_TIME := as.POSIXct(RECORDED_TIME,
  format = "%Y-%m-%dT%H:%M:%SZ", tz = "UTC")]

# Identify all instances of a drop of 40 over 6 hours
value_change(systolic_bp, value = 40, direction = "down", window_hours = 6,
 join_key = "PAT_ID", time_var = "RECORDED_TIME",
 value_var = "VALUE", mult = "all")
# Identify first instance of a drop of 40 over 6 hours
```

```
value_change(systolic_bp, value = 40, direction = "down", window_hours = 6,
 join_key = "PAT_ID", time_var = "RECORDED_TIME",
 value_var = "VALUE", mult = "first")
# Identify last instance of a drop of 40 over 6 hours
value_change(systolic_bp, value = 40, direction = "down", window_hours = 6,
 join_key = "PAT_ID", time_var = "RECORDED_TIME",
 value_var = "VALUE", mult = "last")
# Identify all instances of drops and increases of 40 over 6 hours
value_change(systolic_bp, value = 40, direction = "all", window_hours = 6,
 join_key = "PAT_ID", time_var = "RECORDED_TIME",
 value_var = "VALUE", mult = "all")
```

`vitals` *Synthesized vital sign measurements for cohort of 100 synthetic patients*

Description

A dataset containing 35,146 lab results for 100 unique synthetic patients. There are 4 unique vital signs, including systolic blood pressure, heart rate (pulse), respiratory rate, and temperature. Patient-level sampling rate, mean numeric value, standard deviation, and number of measurements per patient follow patterns observed in real electronic health record data. This dataset is included to demonstrate sepsis identification.

Usage

```
vitals
```

Format

A data frame with 35146 rows and 4 variables:

PAT_ID patient identification number, randomly generated integer

RECORDED_TIME recorded datetime of vital sign measurement, character

VALUE vital sign measurement value, numeric

VARIABLE categorical variable specifying the vital sign name, character

Source

Randomly synthesized patient data

Examples

```
## Not run:
 vitals

## End(Not run)
```

Index